National Aeronautics and
Space Administration

NSTS 07700–10–MVP–09
REVISION B
AUGUST 26, 1993

Lyndon B. Johnson Space Center
Houston, Texas 77058

REPLACES
NSTS 07700–10–MVP–09
REVISION A

SPACE SHUTTLE

SHUTTLE MASTER VERIFICATION PLAN

VOLUME IX
COMPUTER SYSTEMS AND SOFTWARE VERIFICATION PLAN

PART I
GUIDELINES AND STANDARDS

REVISION LOG

REV LTR	CHANGE NO	DESCRIPTION	DATE
		BASELINE ISSUE (Reference: Level II PRCBD S01751A)	1/07/76
A	1	REVISION A (Reference: Level II PRCBD S40129, dated 7/23/86)	9/10/86
		REISSUE (Reference: Notice in front of document) including Changes 1 and 2.	1/20/89
B	5	REVISION B (Reference: SSP DOC–125, dated 8/11/93) also includes Space Shuttle PRCBDs S004600G, S052730, SSP DOC–106, SSP DOC–123 and Changes 2 thru 4.	8/26/93

CHANGE SHEET

FOR

PROGRAM DEFINITION AND REQUIREMENTS
SHUTTLE MASTER VERIFICATION PLAN
VOLUME IX − Computer Systems and Software Verification Plan
Part I − Guidelines and Standards

CHANGE NO. 7

Program Requirements Control Board Directive Nos. S061424/(1-1), dated 5/2/00; S071024EB, dated 10/27/97 and SSP DOC−425.(1)

May 30, 2000

Robert H. Heselmeyer
Secretary, Program Requirements
Control Board

CHANGE INSTRUCTIONS

1. Remove the following listed pages and replace with the same numbered attached pages:

Page	PRCBD No.
iii	S061424, S071024EB, SSP DOC−425
iv	
1−1	S061424
1−2	

NOTE: A black bar in the margin indicates the information that was changed.

2. Remove the List of Effective Pages, dated January 5, 1995 and replace with List of Effective Pages, dated May 30, 2000.

3. Sign and date this page in the space provided below to show that the changes have been incorporated and file immediately behind the List of Effective Pages.

_____ _____

Signature of person incorporating changes Date

PROGRAM DEFINITION AND REQUIREMENTS
SHUTTLE MASTER VERIFICATION PLAN
Volume IX – Computer Systems and Software Verification Plan
Part I – Guidelines and Standards

*Revision B (Reference PRCBD Nos. S004600G, dated 6/23/93; S052730, dated 7/1/93; SSP DOC–106; SSP DOC–123 and SSP DOC–125)

LIST OF EFFECTIVE PAGES

May 30, 2000

The current status of all pages in this document is as shown below:

Page No.	Change No.	PRCBD No.	Date
i	Rev. B	*	August 26, 1993
ii	6	S004600J S052558E	October 21, 1993, July 29, 1994
iii	7	S061424 S071024EB SSP DOC–425	May 2, 2000, October 28, 1997, April 27, 1999
iv – x	Rev. B	*	August 26, 1993
1–1	7	S061424	May 2, 2000
1–2 – 1–4	Rev. B	*	August 26, 1993
2–1 – 2–24	Rev. B	*	August 26, 1993
3–1 – 3–6	Rev. B	*	August 26, 1993
A–1 – A–8	Rev. B	*	August 26, 1993

NSTS 07700–10–MVP–09
PART 1

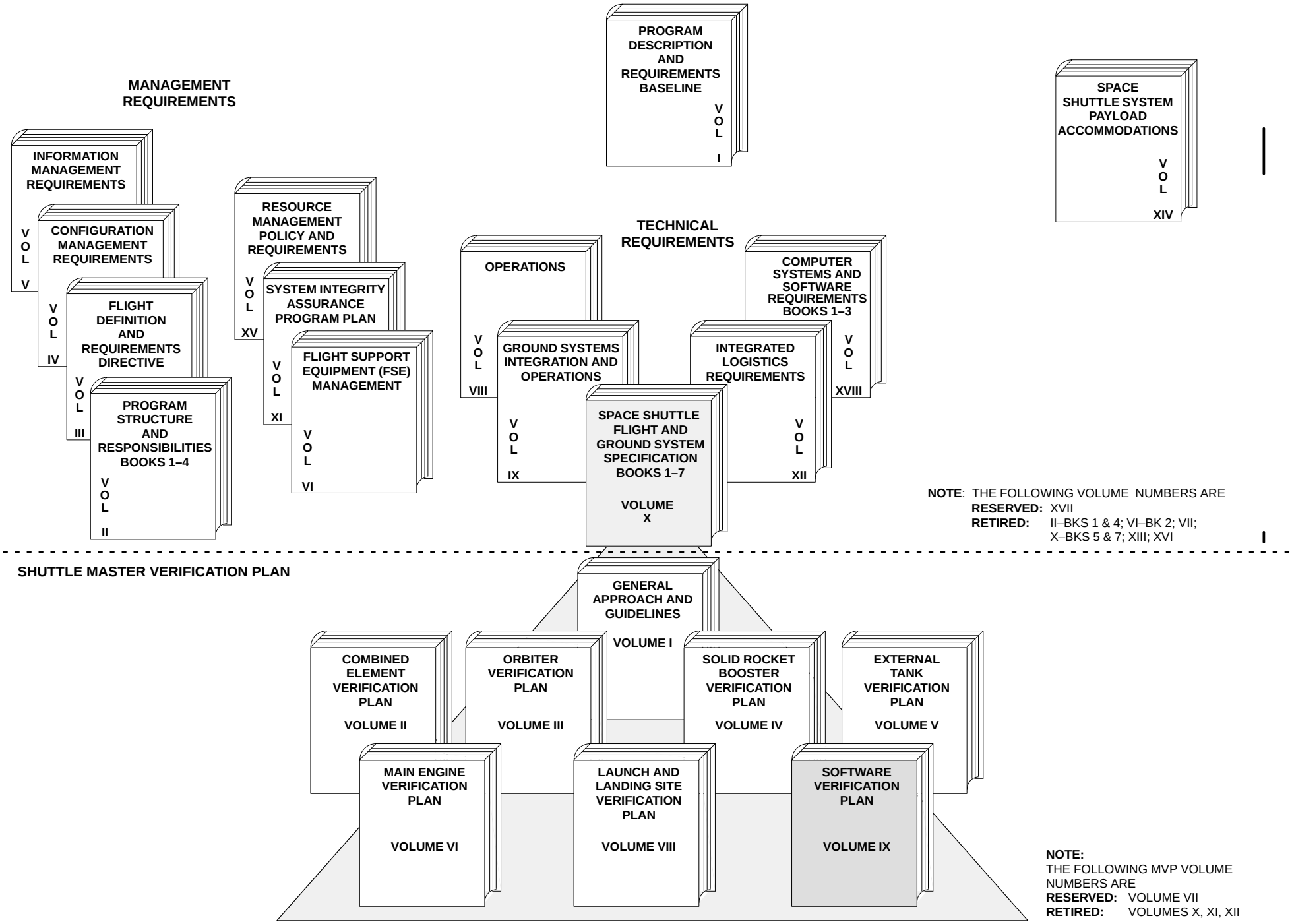
SPACE SHUTTLE

SHUTTLE MASTER VERIFICATION PLAN

VOLUME IX
COMPUTER SYSTEMS AND SOFTWARE VERIFICATION PLAN

PART I
GUIDELINES AND STANDARDS

SPACE SHUTTLE
PROGRAM DEFINITION & REQUIREMENTS – NSTS 07700



NOTE: THE FOLLOWING VOLUME NUMBERS ARE
RESERVED: XVII
RETIRED: II–BKS 1 & 4; VI–BK 2; VII; X–BKS 5 & 7; XIII; XVI

NOTE:
THE FOLLOWING MVP VOLUME NUMBERS ARE
RESERVED: VOLUME VII
RETIRED: VOLUMES X, XI, XII

FOREWORD

Efficient management of the Space Shuttle Program (SSP) dictates that effective control of program activities be established. Requirements, directives, procedures, interface agreements, and system capabilities shall be documented, baselined, and subsequently controlled by SSP management.

Program requirements, directives, procedures, etc., controlled by the Program Requirements Control Board (PRCB), are documented in the volumes of this document, NSTS 07700. The accompanying illustration identifies the volumes that make up the Space Shuttle Program Definition and Requirements. Volume I contains overall descriptions of the NSTS 07700 documentation. Requirements to be controlled by the NASA project managers are to be identified, documented, and controlled by the project.

Volumes I, II and IX of the Space Shuttle Master Verification Plan are approved by the PRCB. Project verification plans documented as Volumes III through VI and VIII are approved and controlled by the respective NASA project offices. Project volumes are maintained as directed by the respective project office.

Volume IX of the Shuttle Master Verification Plan contains the Shuttle Program Computer Systems and Software Verification Plan. Part I of this volume identifies the guidelines and standards for verification of major computer systems and software, and Part II identifies the SSP computer system integration verification requirements. The Office of Primary Responsibility (OPR) for NSTS 07700–10–MVP–09 is the Avionics and Software Office.

All elements of the SSP must adhere to these baselined requirements. When it is considered by the Space Shuttle Program element/project managers to be in the best interest of the SSP to change, waive or deviate from these requirements, an SSP Change Request (CR) shall be submitted to the Program Requirements Control Board (PRCB) Secretary. The CR must include a complete description of the change, waiver or deviation and the rationale to justify its consideration. All such requests will be processed in accordance with NSTS 07700, Volume IV, and dispositioned by the Manager, Space Shuttle Program, on a Space Shuttle PRCB Directive (PRCBD).

_____

Ronald D. Dittemore
Manager, Space Shuttle Program

THIS PAGE INTENTIONALLY LEFT BLANK

CONTENTS

CONTENTS

APPENDICES

TABLES

FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 INTRODUCTION

1.1 PURPOSE

The Space Shuttle Program Computer Systems and Software Verification Plan has two objectives: (1) to provide a consistent and systematic set of guidelines and standards for testing Shuttle Program software systems and (2) to provide specific requirements for integration verification of major Shuttle computer systems. To satisfy the first objective, this document includes a description of the guidelines and standards, verification processes, documentation, and controls which provide management visibility into the verification of Shuttle computer systems and software. To satisfy the second objective, Part II of the plan identifies Space Shuttle Program (SSP) integrated computer systems hardware/software verification requirements, test specifications, and acceptance criteria.

1.2 SCOPE

MVP–09, Part 1 is applicable to Space Shuttle Program computer systems (as defined in Section 2.1). All computer systems and complexes which directly support the development and testing of the Space Shuttle system or inter–project deliverables will adhere to this baselined document. Where it is considered that the requirements should be waived, deviated from, or changed, the proper waiver, deviation, or change request accompanied by a detailed justification and explanation of the alternative procedures must be submitted to the proper management level in accordance with established procedures.

1.3 MASTER VERIFICATION ORGANIZATION

The Computer Systems and Software Verification Plan is Volume IX of a set of documents entitled Master Verification Plan (the figure accompanying the FOREWORD of this document presents an overview of the SSP requirement documents and other volumes of the Master Verification Plan). This volume of the Master Verification Plan (MVP) defines the verification standards and requirements which are applicable when the purpose of a test involves the verification of computer systems and software. Higher level system or element testing which happens to include computers among other “subsystems” will also use MVP Volumes III through VI and VIII as sources for test requirements.

Volumes I, II and IX of the MVP are approved and maintained by the PRCB. Shuttle Element Verification Plans (Volumes III through VI and VIII) are maintained, approved, and controlled by the respective NASA project/element offices. The documents are briefly described below:

SPACE SHUTTLE PROGRAM

Volume I – General Approach and Guidelines

Introduces the overall plan, describes the approach to Shuttle system verification, and provides the verification program guidelines required to be applied throughout the Shuttle System. It also identifies the assigned program responsibilities, the documentation requirements, and the control of Shuttle Program verification requirements. A summary of the test program is included.

Volume II – Combined Element Verification Plan

Identifies the combined element and system–level verification requirements and the methods established for verification of each requirement. It also describes the analysis and test programs to be conducted at the Shuttle system level and on other configurations that incorporate two or more elements.

Volume IX – Computer Systems and Software Verification Plan

Provides a consistent and systematic set of guidelines and standards for testing Shuttle Program software systems (Part I). It also identifies specific requirements for integration verification of major Shuttle computer systems (Part II). Appendix A presents a glossary of terms which have specialized meaning in the area of computer systems and software verification.

Volume X – Retired

PROJECTS

Volume III, IV, V, and VI – Element Verification Plans

Contain element–level requirements and planning information. They are prepared by each element contractor and consist of development, qualification, analysis, and test plans required to provide element verification. Volumes III, IV, V, and VI are for the Orbiter, Solid Rocket Booster (SRB), External Tank (ET), and Main Engine (ME), respectively.

Volume VII – Reserved

Volume VIII – Launch and Landing Site Verification Plan

Establishes the requirements and plans for verification of the Kennedy Space Center Launch and Landing Site as a major program element. It treats those verification activities which must be accomplished to assure readiness of the Ground System to support the flight elements.

1.4 RELATIONSHIP TO SPACE SHUTTLE PROGRAM REQUIREMENTS (NSTS 07700 VOLUME XVIII)

Book 3 of Volume XVIII shall be the source of requirements for documentation and milestone reviews relating to computer systems and software development, including those

governing testing. Some of these will be repeated in MVP–09, Part 1 (Sections 1 and 2), for clarity and continuity of the requirements specified in this volume.

1.5 APPLICABLE DOCUMENTS

The following documents of the date and issue shown form a part of this document to the extent specified herein. "(Current Issue)" is shown in place of a specific date and issue when the document is under Space Shuttle PRCB control. The current status of documents shown with "(Current Issue)" may be determined from NSTS 08102, Program Document Description and Status Report.

NSTS 07700 Volume XVIII Book 3 (Current Issue)	Computer Systems and Software Requirements, Software Management and Control
	Ref. Para. 1.4, 2.2.2.2, 3.0
NSTS 07700–10– MVP–01 (Current Issue)	Shuttle Master Verification Plan – General Approach and Guidelines, Volume I
	Ref. Para. 1.3, 2.2.1.5
NSTS 07700–10– MVP–02 (Current Issue)	Shuttle System Master Verification Plan – Combined Element Verification Plan, Volume II
	Ref. Para. 1.3, 2.2.1.5, Fig. 2–2, 2–4
JSC 07700–10– MVP–03	Orbiter Verification Plan, MJ072–0004–3, Volume III
	Ref. Para. 1.3, 2.2.1.5, Fig. 2–2
JSC 07700–10– MVP–04	Solid Rocket Booster Verification Plan, Volume IV
	Ref. Para. 1.3, 2.2.1.5, Fig. 2–2, 2–4

JSC 07700–10– MVP–05	External Tank Verification Plan, MMC–ET–TM01–B, Volume V Ref. Para. 1.3, 2.2.1.5, Fig. 2–2, 2–4
JSC 07700–10– MVP–06	Main Engine Verification Plan, DVS–SSME–NNN, Volume VI Ref. Para. 1.3, 2.2.1.5, Fig. 2–2, 2–4
JSC 07700–10– MVP–08	Launch and Landing Site Verification Plan (KSC–K–STSM–09, Volume IV – Supplement) Ref. Para. 1.3, 2.2.1.5, Fig. 2–2, 2–4
NSTS 07700–10– MVP–09 Part II (Current Issue)	Shuttle Master Verification Plan – Computer Systems and Software Verification Plan, Verification Requirements, Volume IX Ref. Para. 1.1, 1.3, Fig. 2–2, 2–4

2.0 DEFINITIONS AND OVERVIEW

Because of the large number of computer systems involved in the overall Shuttle Program, it is important that a correct and common basis of test related activities be clearly established. Common verification standards and management and control mechanisms will assure the effective use of Shuttle Program resources and aid in meeting milestones. Section 2.0 presents the basic groundrules and rationale used to develop the detailed guidelines and standards in Section 3.0.

2.1 COMPUTER SYSTEMS AND SOFTWARE VERIFICATION

A computer system is defined as either (1) an electronic system for which the control mechanism is principally a digital computer or (2) a computational facility consisting of a computer and software programs which support a development or operational activity. Software is defined as the entire set of instructions and data which are executed within the environment of a computer system.

Computer systems and software verification is defined as the complete process of ensuring that the software programs and the computer systems satisfy all design requirements, and the means by which the test requirements are satisfied. The verification process and the verification guidelines and standards as described in detail in Paragraph 2.2 and Section 3.0, respectively, are based on the following fundamental groundrules.

2.1.1 Parallel Design and Test Planning

Test requirements, plans, and procedures shall be identified by an independent test function as the software design effort is progressing. The purpose of these parallel design and test activities is to ensure that any possible impact testing has on design (built in testability) is identified early in the design process, that adequate test planning exists such that test facilities have sufficient build up time to support the testing, and that the design requirements are verified. Figure 2–1 shows the parallel activities and the systematic test sequence.

2.1.2 Systematic Test Sequence

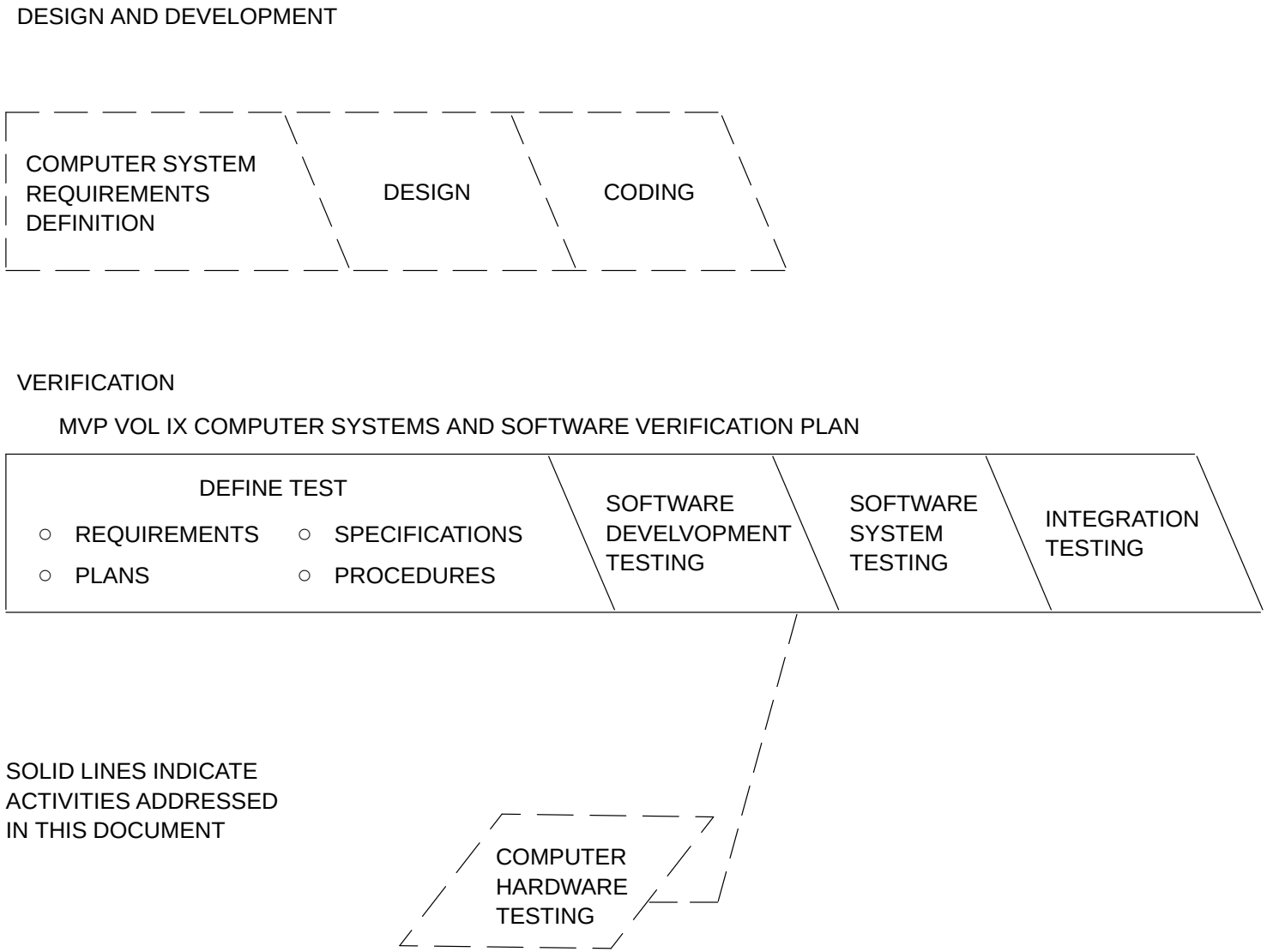
The verification of computer systems and software shall conform to a systematic test sequence in which each subsequent test is a progressive extension of previous tests. Each test (including retesting of modified software) shall make effective use of previous test data and will involve the verification of increasingly more interfaces until the complete program verified. This program (with an associated computer) is then tested with interfacing subsystems. Upon successful completion of these tests, integration testing between computer systems then proceeds. This systematic test sequence concept

also implies that effective interface between design, prototype, and production systems/ software prior to integration testing is a part of this logical test sequence. Paragraph 2.2.1 details the systematic test sequence for computer systems and software.

Much of what is set forth in Section 2.0 of this document is based on the assumption that software is developed as an integral part of a “deliverable” system which evolved in an “end–to–end” manner. There is, however, an important category of programs having special characteristics: namely, operational programs which are written by a user in an interactive on–line fashion, usually in a high–order language. The term “operational” is used to indicate the “end–result” configuration of such programs in distinction to the “deliverable” code which may be generated in a similar manner. A prominent example of on–line operational programming is that planned effort at KSC which will be concerned with generating and/or modifying GOAL–language ground checkout programs for the Orbiter and other Shuttle elements. It should be recognized that such programs (and the systems which support them) impose some unique testing considerations. First, the testing of the on–line facility and its interfaces must be unusually extensive in order to approximate and verify the operational contingencies of a realtime multi–user environment. Second, the operational programs themselves must be verified in the process of their on–line development. The former category of testing is governed by the systematic test sequence concepts discussed above and in Figure 2–1; but the latter is equivalent to an additional level of testing with its own unique requirements, and will be treated as such in later sections of this document.

FIGURE 2–1

SYSTEMATIC TEST SEQUENCE FOR COMPUTER SYSTEMS AND SOFTWARE VERIFICATION



THIS PAGE INTENTIONALLY LEFT BLANK

2.1.3 Top–Down* Modular Design/Test Concept

A software module is defined as the smallest software package which functions as a component of a software program. Cost effective development and verification of computer systems and software involve the application of the computer program modularity concept. The “top–down” approach to software verification shall be the normal testing process employed with the modular software concept and the systematic sequence as described previously. Top–down testing emphasizes a hierarchical order of testing the top level control (executive and supervisory) software and progressing to lower level software after the higher level has been verified. The top–down approach to software design and testing begins with the overall program structure and letting the terminal points of the program be represented as pseudo modules (sometimes referred to as stubs or dummy modules). The pseudo modules each have the major characteristics of the actual end–product modules, (i.e., input and output characteristics, etc.). Top–down testing involves the use of these pseudo modules to represent the lower interfaces required to exercise the higher level software. The pseudo modules are replaced with the actual modules (as they are developed) once the higher level control software has been verified.

* In the expression “top–down” it is assumed that “top” refers to the highest level of control within the system hierarchy.

It may be cost effective to test certain software modules off–line with a dummy driver before testing the module and its interfaces on–line (i.e., top–down). It is expected that dummy drivers will be used for testing unproven algorithms and formulation verification studies. Dummy drivers may also prove cost effective where:

a. Exhaustive testing of modules requires significant computer time using the true driver.

b. Schedule considerations require parallel testing of many modules.

c. Computer hardware availability problems exist.

d. It is difficult to select input values at systems levels to execute particular branches or options within a low level module.

When determining the cost effectiveness of using dummy drivers, the cost of developing the driver and the cost of potentially additional testing using the true driver (executive and supervisory) should be considered.

2.1.4 Documentation and Control

Documentation and controls shall be established to ensure visibility into the verification of computer systems and software. Verification documentation shall address the following activities:

a. Management and Test Development Planning

b. Quality Assurance Requirements

c. Test Requirements

d. Test Planning

e. Test Specifications

f. Test Procedures

g. Test Evaluation

Effective product control procedures (configuration management) shall be established. Milestone reviews shall be supported by the appropriate documentation and spaced throughout the complete verification process to ensure that interfacing system or element milestones are met. The specific documents and reviews which relate to these activities are discussed in Paragraph 2.2.2.2.

2.2 VERIFICATION PROCESS

Verification, as used in this volume, refers to the demonstration of the successful implementation of a design requirement. There are two acceptable methods for the verification of computer systems and software: test and analysis. The test method is the primary method of software verification. The test method of software verification is defined as the exercising of a given computer program's actual code (or instructions and data).

2.2.1 Levels of Testing

Figure 2–2 presents a summary of the levels of computer systems and software testing. Each level of testing is directed toward a specific verification objective. The division of the verification process into a logical set of discrete levels of testing (based upon the "systematic test sequence") provides the basis for management visibility and control, as well as standard objectives for the developers and testers. The discussion of levels of testing presented in the following paragraphs pertain to computer systems and software verification in general. It is understood that the exact techniques of software testing will be adapted to the nature of the system being developed. The order in which the levels are discussed does not necessarily dictate the chronology of testing.

2.2.1.1 Formulation Verification

Formulation verification is performed upon a portion of a computer program to establish software design methods or performance acceptability. It is during this level of testing

that new or unproven algorithms and numerical methods are subjected to experimental test and analysis methods. The need for these tests should be specified at the Design Requirements Review (DRR).

Where a new algorithm or method is to be used in a critical software program, the results of these formulation verification tests shall be documented in Formulation Verification Analysis Reports. These reports shall be available for inputs to the milestone summary reports.

2.2.1.2 Module Testing

Module testing refers to the verification of the lowest level of software functions within a program. As each module becomes verified in a top–down approach, it shall be baselined and placed under configuration control (Paragraph 3.6). Each of the controlled modules should subsequently be treated as a “sealed unit” with known inputs and outputs for verifying their interactions and interfaces with other unit modules as they replace their dummy modules.

Module testing involves using selected inputs, executing the code, and reviewing produced outputs. Module testing is satisfactorily completed when correct results are produced from inputs which are in the proper range and when the intended function of the module is verified. The lowest level software modules are exhaustively verified. That is, test cases are designed to (1) exercise all reasonable branches and executable statements, (2) verify that, for the range of variables over which the software is designed, there are no inherent software errors, and (3) verify the functional requirements. Module testing shall begin after completion of, and in response to, the following documentation:

a. Software Test Requirements

b. Software Test Plan

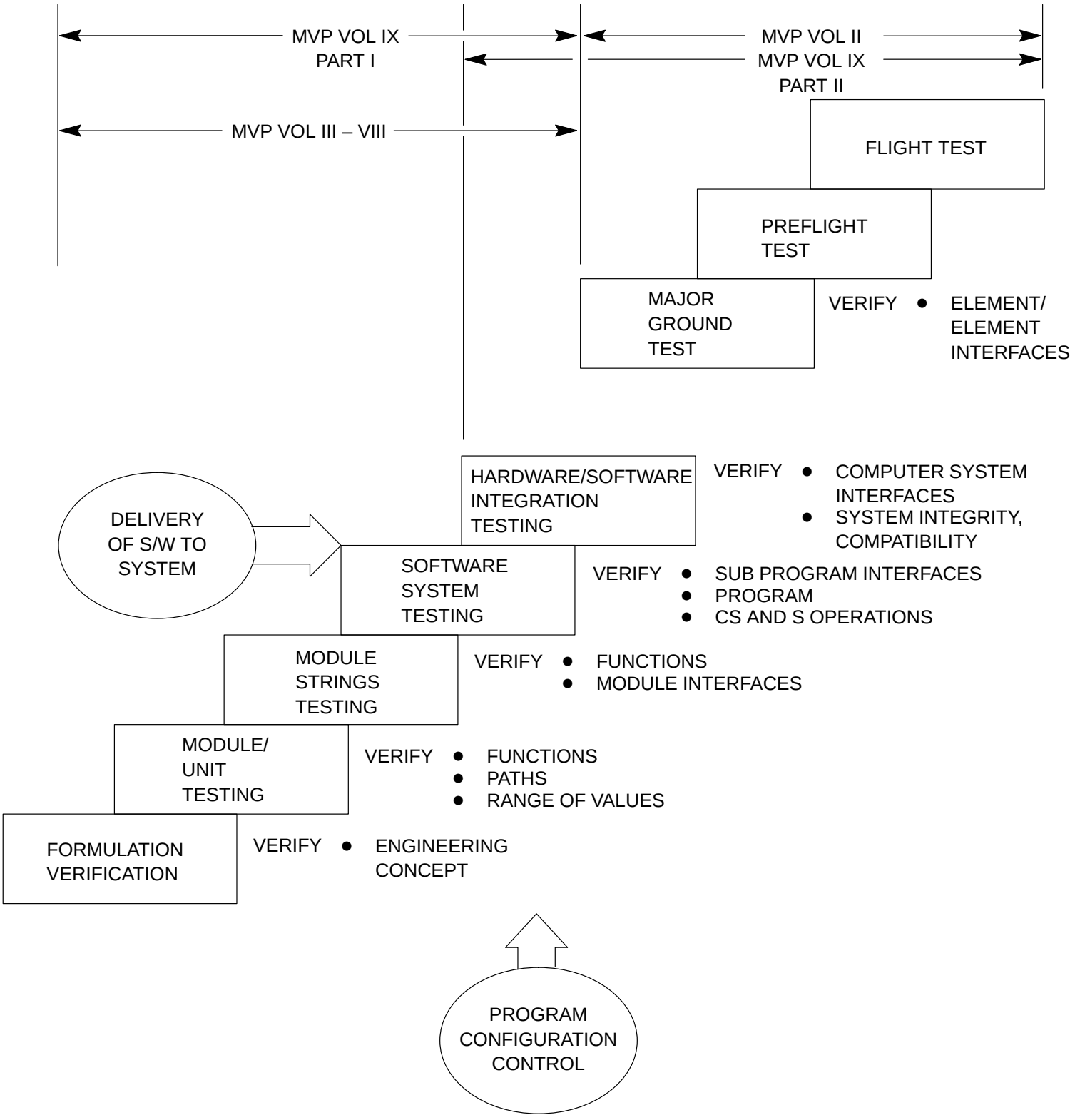
c. Software Development Test Specification

This documentation plus the Module Test Report document and a set of test data shall be retained for subsequent comparisons and/or modifications.

THIS PAGE INTENTIONALLY LEFT BLANK

FIGURE 2–2

VERIFICATION PROCESS – LEVELS OF TESTING



THIS PAGE INTENTIONALLY LEFT BLANK

2.2.1.3 String of Modules Testing

As the individual modules are verified and able to stand alone, emphasis is placed on properly verifying the interfaces between combinations or strings of modules. Each string is tested to assure that proper interfaces exist.

2.2.1.4 Software System Testing

During software system verification, the various combinations of strings and modules are being verified for proper interfacing and interaction. This level of test, as well as the module interface testing in the previous level, must ensure interface compatibility. This requires that the data range of output parameters be within the expected range with respect to the input parameters, the calling sequence is compatible, and the data stored in a common data base is consistent with data read from that data base.

The software/system verification activity is primarily concerned with the deliverable computer program. It is at this test level that the software must be completely verified within its own operating system. The total software package, operating with the specific computer, will have been completely verified when the software is “delivered to the system” (for example, Avionics Data Processing System, Main Engine Controller, etc.) at the formal Configuration Inspection (CI). “The system” may be a high fidelity laboratory simulation of the flight system, like the Shuttle Avionics Integration Laboratory (SAIL).

In addition to the documents mentioned above, the “software system test specifications” must be documented prior to commencing software/system testing. The “Software Test Report” must be available to support the formal CI. As is the case for all computer systems and software testing, the requirements, plans, specifications, and test report documentation plus a set of test data must be retained for future comparisons or modifications.

2.2.1.5 Hardware/Software Integration Testing

The purpose of integration testing is to verify the operation of the computer system (and software) with its immediate interfaces. These tests shall include not only the verification of the computer system with interfacing cabling, sensors, displays, and keyboards, but also other computer systems or major electronic systems.

Table 2.1 shows the guidelines for establishing the integration test sequence. These guidelines are based on the previous groundrules and the following objectives:

a. Early verification of interface related software

b. Verification that the computer system is ready for integration testing

c. Integration testing for interface verification of the operational system.

The degree to which the guidelines for selecting the integration test sequence are applied to each interface shall be based on design confidence and cost. The table shows the time phasing of integration testing. The final integration testing will be performed with the flight or operational version of the program. In the case of test facility computer systems, integration testing shall include those computer programs which support the facility's operational function.

Shuttle operational computer systems interface verification has been divided into two parts. Most integration testing will be accomplished at the project level and the verification requirements are included in MVP Volumes III through VI and VIII. Those interfaces which involve Element Interfaces are defined to be at the program level and the verification requirements are contained in Part II of this volume.

In the event that certain functions may be more economically tested during major ground test* with minimum risk, each developer/tester must ensure that these requirements are documented as inputs to higher level requirements to ensure that all functions are tested. Such deferrals of element test requirements are subject to program level concurrence.

* Major Ground Test is defined in the Master Verification Plan (Volume 1 and 2) and involves the testing of major hardware systems at test facilities as opposed to "flight test".

2.2.2 Generalized Responsibilities, Documentation and Controls

This section defines the generalized responsibilities of the management, development, test and test facility, and quality assurance functions for the verification of computer systems and software. The controls and documentation required in performing these responsibilities are also defined. The responsibilities, documentation, and controls identified in this section are based on the modular software design, top–down test concept and systematic test sequence described in the previous sections. The application of these concepts results in common objectives for the responsible test organizations and provides a means by which management can measure progress and take corrective actions when activities deviate from plans.

2.2.2.1 Generalized Responsibilities

An effective test program requires the identification of the roles and responsibilities of the organizations contributing to the completion of computer systems and software. Experience indicates the organizations must accomplish four functions for effective software development and testing.

a. Management

b. Design/Development

c. Test and Test Facilities

d. Quality Assurance

The delivery of valid computer systems and software on schedule and within budget is the responsibility of the management function. The design/development function is directed by the management function and is responsible for defining the design and development of the software plus the testing of the software through the unit module level. The test function is responsible for the verification of the design/development function. The principal concern in the test function is to demonstrate compliance of the product with the requirements imposed by the development specification. If the requirements of the development specification are not testable or realistic, the test function will initiate requirement reevaluations. The independent test function shall also review the design while preparing comprehensive test plans and test procedures for validation testing. This process also ensures a deeper analysis of the testability of the computer programs and encourages the documentation of interface requirements early in the design phase.

The test facility shall be responsible for providing the capability to test the requirements identified by the test function. Fidelity of the test facility capability shall be established consistent with acceptance criteria of the test requirements.

The purpose of the Quality Assurance (QA) function is to audit (and report) the design/development and test functions. QA shall be responsible for ensuring that testing standards and approved procedures are followed.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE 2.1

GUIDELINES FOR SELECTING THE INTEGRATION TEST SEQUENCE

	SYSTEMATIC TEST SEQUENCE		
	INTERFACE DESIGN CONCEPT TESTING	PRELIMINARY (OR PARTIAL) INTEGRATION TESTING	FINAL INTEGRATION TESTING
OBJECTIVES	VERIFICATION OF INTERNAL OPERATIONS WHICH AFFECT INTERFACE	VERIFY SYSTEM WITH IMMEDIATE INTERFACE OR PROTOTYPE INTERFACE STRINGS	VERIFY SYSTEM–TO–SYSTEM INTERFACE
TYPES OF TEST REQUIREMENTS	• VERIFY CAPABILITY TO TRANSMIT • VERIFY CAPABILITY TO RECEIVE	• VERIFY CAPABILITY TO RECEIVE SIGNAL FROM INTERFACE SYSTEM OR SENSOR • VERIFY COMMAND RESPONSE	• VERIFY COMPUTER SYSTEM–TO–COMPUTER SYSTEM INTERFACE • VERIFY COMPUTER SYSTEM–TO–ELEMENT SYSTEM INTERFACE
TYPES OF ACCEPTANCE CONSIDERATIONS	• SIGNAL CHARACTERISTICS • FORMAT • RATES	• PROPER RESPONSE • RECEIVED DATA	• PROPER SEQUENCE OF RESPONSES
CONFIGURATION	• PRELIMINARY (BREAD BOARD) SYSTEM DESIGN • DUMMY INTERFACES • OPEN LOOP	• PREPRODUCTION SYSTEM TO SYSTEM • SECOND LEVEL INTERFACES MAY BE DUMMY • OPEN–LOOP WITH LIMITED CLOSED–LOOP	• PRODUCTION SYSTEM–TO–SYSTEM INTERFACE • CLOSED–LOOP TESTING WITH SIMULATED FLIGHT COMPONENTS (THRUST, ETC)
APPROXIMATE TIME PHASING	CRITICAL DESIGN REVIEW	CONFIGURATION INSPECTION	ACCEPTANCE REVIEW

The specific responsibilities of each of these functions are given below.

Management Function

a. Establish management and test
   – Responsibilities
   – Schedules
   – Documentation
   – Reviews
   – Change Control

b. Requirements

c. Waivers to standards

d. Design changes during testing
   – Discrepancy reporting
   – Software acceptance/delivery criteria

e. Establish development, test and quality assurance organizations

f. Establish control boards and panels

g. Establish mechanisms for integrating development, test and user requirements (panels, as required)

h. Approve test requirements
   – Responsibilities
   – Acceptance criteria
   – Traceability
   – Test facilities
   – Schedules and milestones

i. Approve quality assurance plan

j. Approve special studies for requirements (formulation verification)

k. Approve test plans

l. Approve test specifications

m. Conduct formal milestone reviews and periodic status reviews

n. Approve input test requirements to subsequent higher level tests

o. Assure the effective completion of the tests with valid results

p. Approve test reports

Development Function

a. Responsible for testing through the unit module level

b. Control configuration of software

c. Document formulation verification and module test results

d. Support test organizations

e. Document design review summary reports

f. Assess impact on schedule and cost of discrepancies identified by the test function

g. Perform software modifications as identified in approved change requests. The software developer will provide timely support for making minor modifications to software at the test facility site if feasible from a cost/schedule standpoint

Test Function

a. Identify and document test requirements

b. Schedule and verify test requirements

c. Review formulation verification documents

d. Coordinate test with developers and test facility

e. Control test configuration

f. Establish and document test plans

g. Establish and document test specifications

h. Define and document test procedures

i. Conduct tests

j. Analyze test results

k. Report discrepancies and recommend design changes

l. Retest modified software

m. Document the test reports

n. Identify inputs to subsequent higher level tests

o. Release test documents and data to library

Test Facility Function

a. Provide capability to satisfy test requirements

b. Coordinate test with test and quality assurance organizations

c. Provide documentation as defined in the management plans or test plans

d. Operate test facility

e. Record and disseminate test data

Quality Assurance Function

a. Participate on CM Board and panels

b. Establish quality assurance plan

c. Verify and ensure test configuration

d. Ensure test standards and approved procedures are followed

e. Document nonconformances during conduct of test

f. Document results of each test

g. Ensure proper configuration of released end items

h. Report discrepancy status at formal reviews

i. Maintain and control library of test documentation and data

j. Ensure release of deliverable end items

2.2.2.2 Documentation and Reviews

Appropriate and timely documentation is required to track and control computer systems and software verification. Those organizations responsible for verification of computer systems and software shall address themselves specifically to the documentation requirements of Software Management and Control, NSTS 07700, Volume XVIII (Book 3), and to the additional requirements contained in this section.

Figure 2–3 depicts the inter–relationships of the major documents which are instrumental in the verification of software and computer systems. A detailed description of the objectives of most of these documents is contained in Volume XVIII (Book 3). A description of the documents unique to this volume of the MVP is presented in the following subsections.

Figure 2–4 presents the general flow of the majority of the verification testing documents and places each document in an approximate relationship to the software milestone reviews. Most documents are depicted as being prerequisites to reviews rather than rigidly specifying that they be available at a formal design review. Also presented (in bold outline) are the principal verification testing activities under the direct purview of MVP IX.

Documents may be published incrementally. For example, it would be most advantageous to complete the test report for module testing at the time that activity is nearing completion (particularly for large software development tasks). Also, some documents may advantageously be combined. The names of documents may also deviate from those established here. The subject material, however, shall be consistent with the specified requirements.

For purposes of MVP Volume IX, primary emphasis at each milestone review is concentrated on the test related documentation as shown in Figure 2–4, depending upon the complexity and type of software. For example, the flight software will require a Flight Readiness Review (FRR) while most other computer programs will not require this specific review milestone.

Figure 2–5 summarizes the purposes of each milestone review as defined in Volume XVIII (Book 3) and identifies the actions relative to verification testing.

2.2.2.3 Integration Testing Documentation Responsibility

The set of documents pertaining to integration testing is indicated in Figures 2–3 and 2–4 to emphasize the need for such documentation. Because of the nature of integrated testing, the responsibility for producing such documentation will vary from development to development, depending on organizational structure as well as system structure. All included hardware and software development organizations must participate. The lead role responsibility for defining integration testing documentation and assigning and allocating the work of producing it is assumed by the lowest level NASA or Contractor organization whose authority encompasses the “integrated system” in question. For example, the lead role responsibility for integration testing of the Orbiter Avionics System (OAS) mated with the Main Engine Computer belongs to the Space Shuttle Program Office (SSPO), because no lower organization encompasses both elements of the combined system. However, the responsibility for integrating the various computer subsystems within the OAS belongs to the Orbiter Project, and so on. The SSPO may assign certain program integration test objectives to a project, to be accomplished as a part of lower level system integration or development testing. This is desirable when facility schedules, configurations, and resources permit.

THIS PAGE INTENTIONALLY LEFT BLANK

FIGURE 2–3

INTER–RELATIONSHIPS OF VERIFICATION DOCUMENTATION

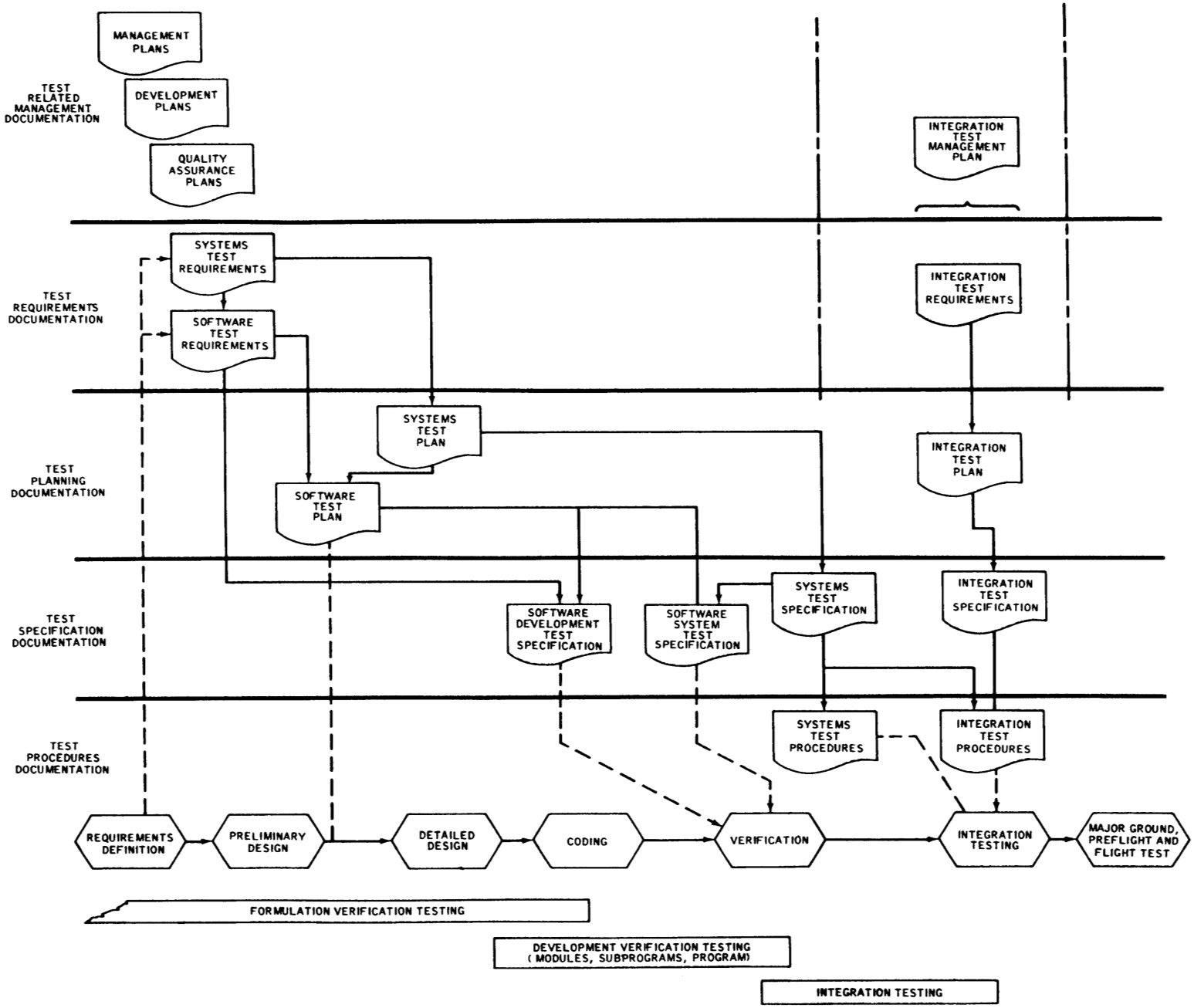


FIGURE 2–4

DOCUMENTATION AND REVIEWS

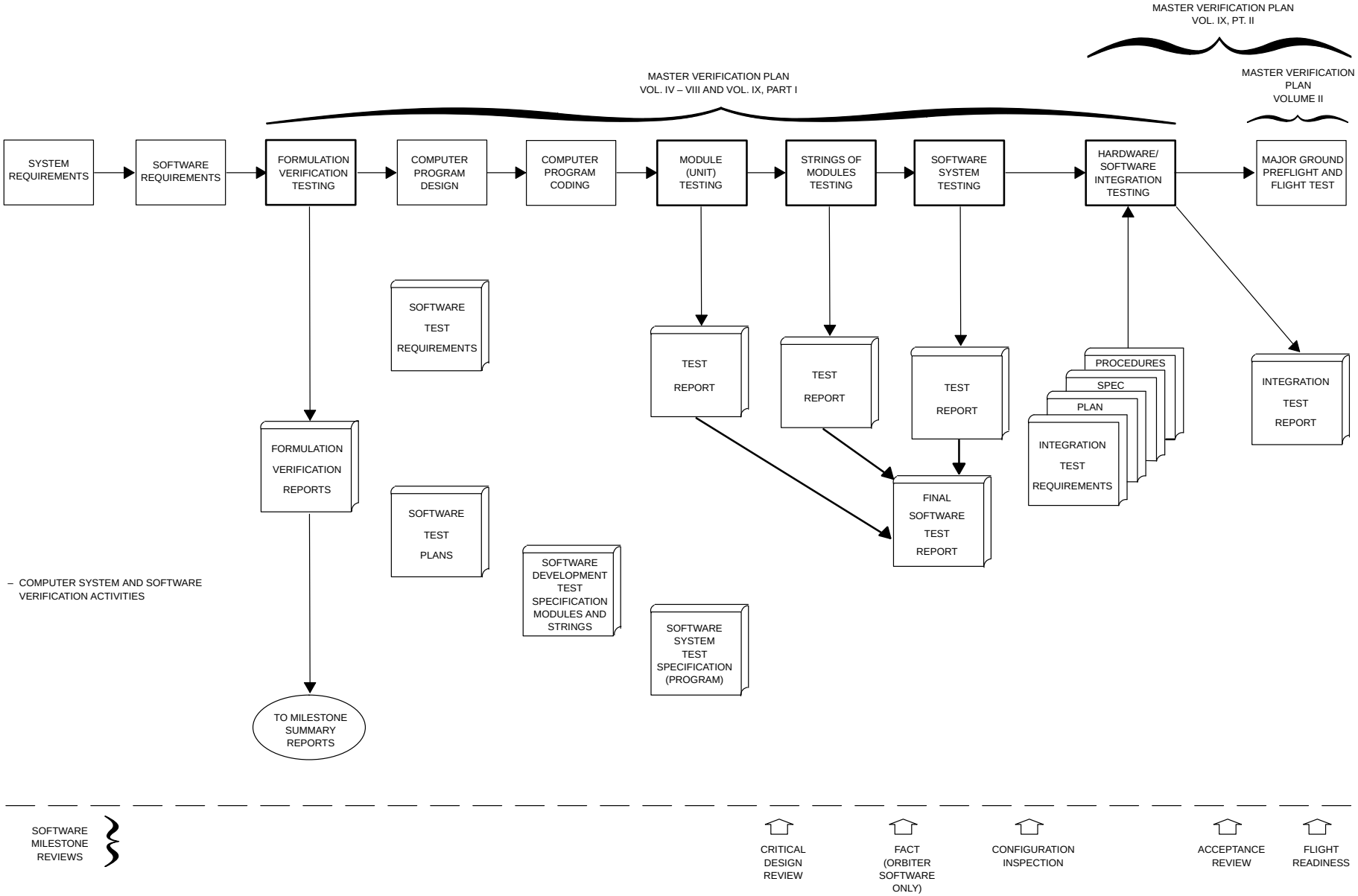


FIGURE 2–5

MILESTONE REVIEWS

(Page 1 of 2)

MILESTONE REVIEW/PURPOSE	PREREQUISITE VERIFICATION ACTIVITIES
SOFTWARE PRELIMINARY DESIGN REVIEW (PDR) ● THE BASELINE ESTABLISHED FOR EACH SOFTWARE PROGRAM AT PDR REPRESENTS A PROPOSED DESIGN APPROACH FROM WHICH THE DETAIL DESIGN PROCEEDS.	● REVIEW THE SOFTWARE TEST REQUIREMENTS TO ESTABLISH THAT THE DESIGN REQUIREMENTS ARE MEASURABLE
SOFTWARE CRITICAL DESIGN REVIEW (CDR) ● THE BASELINE THAT IS ESTABLISHED AT THE COMPLETION OF THE CDR REPRESENTS THE DETAILED SOFTWARE DESIGN AND PROVIDES “CODE–TO” CRITERIA TO DESIGNERS/ PROGRAMMERS SO THAT CODING OF THE SOFTWARE CAN BE FORMALLY AUTHORIZED.	● REVIEW THE TEST REQUIREMENTS, PLANS AND SPECIFICATIONS. ● ESTABLISH THAT THE TEST PLANS AND SPECIFICATIONS DOCUMENTS FOR MODULE TESTING HAVE BEEN PREPARED AND ARE READY TO SUPPORT THE TESTING PHASE OF THE SOFTWARE DEVELOPMENT ACTIVITY.
FIRST ARTICLE CONFIGURATION INSPECTION (FACI) (ORBITER SOFTWARE ONLY) ● DEMONSTRATE READINESS FOR ORBITER SOFTWARE SYSTEM VERIFICATION ● THE ORBITER SOFTWARE PROGRAM AT FACI REPRESENTS PRE–SYSTEM–VERIFICATION ORBITER SOFTWARE	● CONDUCT TEST ● REVIEW OF ORBITER SOFTWARE SYSTEM DEVELOPMENT AND INTEGRATION TEST RESULTS
SOFTWARE CONFIGURATION INSPECTION (CI) ● THE SOFTWARE PROGRAM AT CI REPRESENTS THE SOFTWARE COMPRISING THE “CODE–TO” DESIGN BASELINE PLUS THOSE CHANGES APPROVED BY THE CONFIGURATION CONTROL AUTHORITY AND SUBSEQUENTLY IMPLEMENTED. ● DEMONSTRATE READINESS FOR VALIDATION TESTING.	● CONDUCT TEST ● REVIEW OF SOFTWARE SYSTEM TEST RESULTS

FIGURE 2–5

MILESTONE REVIEWS

(Page 2 of 2)

MILESTONE REVIEW/PURPOSE	PREREQUISITE VERIFICATION ACTIVITIES
<u>SOFTWARE ACCEPTANCE REVIEW (AR)</u> ● ESTABLISH THE ACCEPTABILITY OF THE COMPUTER PROGRAM SOFTWARE ● ESTABLISH THAT THE INTEGRATION TESTING WAS PERFORMED USING THE CURRENT TEST SPECIFICATIONS AND PROCEDURES ● REVIEW TEST RESULTS AND DOCUMENTATION ● REVIEW ALL SOFTWARE CHANGES AND CORRESPONDING TEST RESULTS SINCE CI	<u>FOR HARDWARE/SOFTWARE INTEGRATION TEST</u> ● ESTABLISH PRE–TEST DOCUMENTATION – TEST MANAGEMENT PLANS – TEST REQUIREMENTS – TEST PLANS – TEST SPECIFICATIONS – TEST PROCEDURES ● CONDUCT TEST AND DOCUMENT RESULTS (TEST REPORT) ● REVIEW TEST RESULTS
<u>PERIODIC STATUS REVIEWS</u> ● PROVIDE MANAGEMENT VISIBILITY INTO THE TECHNICAL ASPECTS OF THE COMPUTER SYSTEM AND COMPUTER PROGRAMS ● VERIFY THE TECHNICAL COMPATIBILITY OF THE DIFFERENT WORK ELEMENTS ● ARRANGE FOR TECHNICAL INFORMATION INTERCHANGE	● PREPARE REPORTS AND BRIEFINGS ● COORDINATE TEST AND DEVELOPMENT ACTIVITIES

3.0 GUIDELINES AND STANDARDS

This section presents the computer systems and software verification guidelines and standards. Standards shall be followed by the testing organizations. Guidelines represent suggested methods or alternatives which experience has shown to be acceptable practices. The standards given below are shown in CAPITAL LETTERS. Standards for software management and control are given in Book 3 of Volume XVIII, Computer Systems and Software Requirements (NSTS 07700). These guidelines and standards are consistent with and form the basis for the verification process given in Paragraph 2.2.

3.1 GENERAL

The objective of the computer system and software verification process is to demonstrate and document that the flight, ground operational, and major test programs satisfy specification requirements. This requires that test article and facility computer systems be under configuration control with adequate traceability to specification requirements.

3.2 IMPACT OF VERIFICATION ON SOFTWARE DESIGN

SOFTWARE SHALL BE DESIGNED TO FACILITATE TESTING BY MODULARIZATION AND LOGICAL PROGRAM STRUCTURING, AVOIDING COMPLEX OR UNUSUAL DESIGN AND CODING PRACTICES, AND SIMPLIFYING MODULE INTERFACES. DESIGN REQUIREMENTS SHALL BE STATED WITH SUFFICIENT DETAIL TO PERMIT THE COMPLETE IDENTIFICATION OF VERIFICATION REQUIREMENTS AND ACCEPTANCE CRITERIA.

3.3 INDEPENDENT AND REDUNDANT VERIFICATION

Independent verification (separate development and testing functions) of computer systems and software is an acceptable approach to verification where the overall testing of the system requires major non–developer involvement. Independent verification will be used where the assignment of the verification to the non–developer function is logical and cost effective.

Redundant verification (verification by two separate organizations) is acceptable where management specifies that the risk/criticality of the software justifies the additional cost. Historically, this type of verification has been limited to the verification of critical software such as flight and realtime ground support computer programs.

3.4 TEST REQUIREMENTS, ACCEPTANCE CRITERIA AND TRACEABILITY

TEST REQUIREMENTS AND ACCEPTANCE CRITERIA SHALL BE IDENTIFIED FOR ALL DESIGN REQUIREMENTS, AND SHALL BE DOCUMENTED WITH THE SOURCE OR REFERENCE TO PROVIDE TRACEABILITY TO DESIGN SPECIFICATIONS. Test requirements

shall be identified for all expected ranges of values and conditions under which the software will have to perform. If test requirements or acceptance criteria cannot be defined from the design requirements, the tester will coordinate the requirements with the designer.

3.5 DEBUGGING

Where cost effective, an acceptable method for debugging is having someone other than the coder review the coding (listing) and programming design languages prior to beginning software checkout. This can minimize the number of iterations required to successfully complete an execution of a module by detecting and correcting obvious coding errors. To aid this effort, software programming design languages and subrouting dependency charts will be documented prior to debugging the program. During debugging, optional display of intermediate calculations shall be considered, as will snapshot dumps.

The use of automated diagnostic capability such as execution traces, dumps, and editors to facilitate debugging will be used where cost effective.

3.6 UNIT/MODULE TESTING

UNIT TESTS MUST BE SUFFICIENT TO ENSURE THAT THE UNIT MODULES MAY BE TREATED AS “SEALED MODULES” FOR TESTING WITH INTERFACING SOFTWARE. MODULES WILL BE PLACED UNDER INTERNAL CONTROL BY THE DEVELOPING ORGANIZATION FOLLOWING VERIFICATION. Unit testing must ensure that software modules contain no inherent errors as well as demonstrate that functional requirements are satisfied. Thus, the set of unit test cases for each software module must exercise every reasonable branch and executable instruction. Furthermore, representative test cases must demonstrate that reasonable data values over which the module is designed to operate will not cause a singularity to occur (negative square root, division by zero, loss of numerical significant, etc.).

3.7 STRING OF MODULES AND SOFTWARE SYSTEM TESTING

MODULAR INTEGRATED TESTS MUST ENSURE THAT THE INTERFACES BETWEEN SOFTWARE MODULES OR STRINGS OF MODULES ARE VERIFIED AND THAT THE FUNCTIONAL REQUIREMENTS ARE SATISFIED. The interface testing must ensure that outputs of one module or string of modules are compatible with the required input of another. For example:

a. The data range of the output parameters are within the expected range of the input parameters.

b. The calling sequence is compatible.

c. Data stored in a common data base is consistent with data read from that data base.

3.8 HARDWARE/SOFTWARE INTEGRATION TESTING

SOFTWARE SHALL BE VERIFIED AND PLACED UNDER CONFIGURATION CONTROL BEFORE PROCEEDING WITH THE FINAL INTEGRATION TESTING. The deliverable version of the program will be used in final integration testing. Integration testing will verify the software system interfaces utilizing equipment representative of Shuttle flight and ground systems.

3.9 RETESTING MODIFIED SOFTWARE

MODIFIED SOFTWARE SHALL BE TESTED, AND FULL ADVANTAGE WILL BE MADE OF THE ORIGINAL TEST CASES AND THE SYSTEMATIC TEST SEQUENCE SUCH THAT ONLY THE ORIGINAL TEST CASES AFFECTED BY THE MODIFICATIONS WILL BE RERUN. NEW TEST CASES SHALL BE DEVELOPED TO TEST NEW BRANCHES AND INTERFACES. THE APPROPRIATE LEVEL OF RETESTING WILL BE PERFORMED TO AVOID CREATING NEW UNTESTED INTERFACES. The test data base and available automated tools will be used to facilitate and systematize the retesting process. The results of these tests as well the original test data will be retained in the software library.

3.10 CERTIFICATION AND/OR RETESTING OF “OFF–THE–SHELF” SOFTWARE

A software development organization may choose to use “off–the–shelf” software to satisfy all or part of its requirements. This may range from “low criticality” software, such as standard library routines for simulation purposes, to “high criticality” software, such as operating systems or flight–related compilers. The details whereby such software is certified to a level of confidence consistent with its function will differ from case to case but the following general standards apply:

a. IT SHALL BE THE RESPONSIBILITY OF ANY ORGANIZATION PROPOSING TO PROCURE OFF–THE–SHELF SOFTWARE TO DOCUMENT, PRIOR TO PROCUREMENT, THE PLAN FOR CERTIFYING THAT SUCH SOFTWARE CAN BE ASSIGNED THE SAME LEVEL OF CONFIDENCE WHICH WOULD BE REQUIRED OF EQUIVALENT SOFTWARE OBTAINED THROUGH A “DEVELOPMENT” PROCESS.

b. ONCE CERTIFIED, MODIFICATIONS TO OFF–THE–SHELF SOFTWARE SHALL BE RECERTIFIED ACCORDING TO STANDARDS EQUIVALENT TO THOSE USED FOR THE ORIGINAL CERTIFICATION.

The “off–the–shelf” certification process required above is expected to make maximum use of prior vendor testing results as well as analysis based on actual prior “field usage”

of the software. For less critical software, such data may provide an adequate basis for certification without the need for further formal verification. For more critical software, it is anticipated that extensive testing over several test levels would be required. (Such testing should be included as part of the cost of the off–the–shelf software).

3.11 RETENTION OF TEST DATA

EACH SOFTWARE DEVELOPMENT/TEST ORGANIZATION SHALL ESTABLISH AND DOCUMENT STANDARDS AND AUTHORITIES FOR DATA RETENTION. For successfully completed tests, input data, test procedures, and test results shall be retained to:

a. Provide proof of test requirement satisfaction

b. Eliminate unnecessary retesting

c. Facilitate retesting if modifications are required.

3.12 DOCUMENTATION

Documentation for computer systems and software will include details for each test relating to:

a. Management Plan

b. Test Operations Plan

c. Development Plan

d. Quality Assurance

e. Test Plan

f. Test Specification

g. Test Procedures

h. Test Reports

i. Analysis Report for Formulation Verification

j. Test Facility Documentation

k. Test Requirements for other MVPs

l. Discrepancy Reports

m. Milestone Review Summary Reports

DOCUMENTATION WHICH IDENTIFIES TEST REQUIREMENTS, PLANS, SPECIFICATIONS, AND PROCEDURES SHALL BE FORMALLY DOCUMENTED BEFORE THE PARTICULAR TEST COMMENCES.

Depending upon the complexity of the tests, documents may be combined or published in incremental volumes. Copies of these documents will be available in a library.

3.13 TEST/CHECKOUT SOFTWARE VERIFICATION

There are computer systems being developed within the Shuttle program whose function is to facilitate the performance of test and checkout activities. Typically, these systems provide the user with the capability to generate, load, and execute programs (or modify them) via a higher–order language compiler. Obviously, the “applications software” thus generated is somewhat different from the “system software” which executes the application. The latter should be verified by a formal, phased test sequence, as described elsewhere in this document; the former must be validated by other means, peculiar to the particular system. These means must be documented by the responsible projects(s).

REQUIREMENTS FOR DOCUMENTING THE VERIFICATION OF TEST/CHECKOUT SOFTWARE

a. THE TEST/CHECKOUT SYSTEM DEVELOPER MUST ADDRESS IN HIS TEST PLANS:

    1. THE METHOD(S) FOR ASSURING THAT THE SYSTEM PERFORMS AS REQUIRED WHEN OPERATED OVER A REASONABLE RANGE OF ACTUAL USAGE CONTINGENCIES.

    2. METHOD(S) FOR MAINTAINING THE ABOVE LEVEL OF ASSURANCE WHEN THE SYSTEM IS CHANGED OR WHEN SIGNIFICANT CHANGES IN OPERATIONAL USAGE ARE PLANNED.

b. THE TEST/CHECKOUT SYSTEM USER MUST PROVIDE DOCUMENTATION ADDRESSING THE FOLLOWING:

    1. ASSURANCE THAT THE TEST/CHECKOUT SYSTEM TESTING IS ADEQUATE TO VERIFY THAT SYSTEM PERFORMANCE IS CONSISTENT WITH OPERATIONAL REQUIREMENTS.

    2. ASSURANCE THAT THE USER–DEVELOPED “APPLICATION” CODE OPERATES WITHIN SYSTEM CAPABILITIES AND CONSTRAINTS IN A REALISTIC USAGE ENVIRONMENT.

    3. PLANS FOR INTERFACING WITH THE SYSTEM DEVELOPER TO SUPPORT SYSTEM VERIFICATION AND REVERIFICATION (SEE a–2 ABOVE).

3.14 CONTROLS

The management of each test activity shall establish the necessary reviews, control boards, and panels to assure visibility into all levels of testing and effective completion of the test by meeting milestones within budget. Review milestones (e.g., CDR) may be accomplished incrementally.

APPENDIX A

GLOSSARY

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

GLOSSARY

Acceptance Criterion	The standard of performance that is set for the software or other system set elements to be tested. Normally, this criterion is used to make a pass/fail judgment on the software or other system elements in meeting a requirement.
As–built	The actual configuration of software code.
AR	Acceptance Review. Formal review following integration testing. (See Figure 2–5)
Baseline	A specific configuration of software and/or documentation for which formal change procedures are utilized.
Building Block Concept	The construction of modular software by joining together the smaller individual modular units.
CDR	Critical Design Review. Formal review held prior to coding and module testing. (See Figure 2–5)
Change Control	A system for managing changes to be made in baselined software, involving the orderly consideration of all interests involved in each change. Formal NASA approval is required of all changes having a cost, schedule, or performance impact that is sufficiently significant to cause change in contractual, task, or other agreement with NASA.
CI	Configuration Inspection. Formal review held after the completion of Computer System and Software Testing, and before Integration Testing. (See Figure 2–5)
Code–to	The baselined specification from which the software will be coded.
Combined Elements Testing	Testing of interface between two or more Shuttle elements. These tests follow Integration Testing as defined in MVP IX.
Computer System	Either (1) an electronic system for which the control mechanism is principally a digital computer or (2) a computational facility comprising computer and software programs which support a development or operational activity.

Debugging	The process of detecting, locating, and removing all mistakes from a program.
DRR	Design Requirements Review. First formal review in the design cycle of software development.
Element	See Shuttle Program Element.
Executive	That part of a program which controls the sequencing of subprograms and modules.
FRR	Flight Readiness Review. Final review for flight software.
Functional Path	A string of system and subsystem units by which end–to–end functional operation is accomplished.
Guideline	A statement of what is normally good practice in software development. Guideline documents are not mandatory regulation, but their consideration is strongly recommended.
Independent Verification	The verification of computer systems and software by non–developer organizations (the tester).
Integration	The combining of two or more articles (e.g., modules, strings of modules, programs, system, elements, etc.).
Integration Testing	Verification of computer system to computer system interface or computer system to major electronic system interface.
Interface	Those functional and/or physical relationships between the various hardware, software, and personnel elements that comprise a system which require compatibility for the successful operation of the overall system.
Interpretive Computer Simulation (ICS)	An ICS is a program that performs the simulation of one computer (target computer) on another computer (host computer). All programs written for the real computer can be executed, without modification, on the simulating computer. The contents of each register and memory location simulated will be identical to those of the real computer.
Levels (Testing)	Application of the test buildup sequence to computer systems and software verification. The levels are Formulation Verification, Modules Testing, Strings of Modules Testing, Software System Testing, and Integration Testing.

Main Frame	The central processor of the computer system. It contains the main storage, arithmetic unit, and special register groups. Synonymous with central processing unit. All that portion of a computer exclusive of the input, output, peripherals, and, in some cases, storage units.
Modularity	That quality of the software which results from designing of small independent units (modules).
Module or Unit Module	The smallest software package functioning as a component of the software program.
Module Testing	Verification of unit modules
MVP	Master Verification Plan
NASA Approval	Formally binding NASA approval. The material approved may not be changed without formal action.
Operating System	Integrated collection of service routines for supervising sequences of programs (debugging, input/output, accounting, compilation).
PDR	Preliminary Design Review. Formal review to establish that the preliminary design satisfies requirements, that adequate interfaces are defined, and that the design selected can be implemented. (See Figure 2–5)
Project Office	The NASA organization responsible for the development of a Shuttle element, e.g., Orbiter, Solid Rocket Booster, etc.
Range of Variables or Values	The extremes of data encountered as well as representative values between these extremes.
Redundant Verification	Two independent groups testing the same computer system and software.
Sealed Unit or Black Box	Verified unit modules under configuration control.
Snap Shot Dump	Printout of intermediate program calculations to aid the debugging process.
Shuttle Program Element	A major component of the Shuttle (SRB, Orbiter, ET, SSME, Payload, Launch and Landing).
Software Development	The process of designing, implementing, refining, and documenting new or modified software.
Software End–Product	Software delivered to the external project customer. Also called deliverables.

Software Integration	The process of combining software modules, programs, and data into a complete software program and refining this program.
Software Program	The entire set of instructions and data which are executed within the environment of a computer system.
Software System	All the software utilized by and residing in a computer system.
SRR	System Requirements Review. A formal review held prior to DRR.
String of Modules or Subprogram	A collection of several software modules which performs a major function of the software program.
Structured Program	Application of modularity concept to software program design.
Sub–System	A breakdown of operating packages within a system.
System	An organized collection of hardware, software, and/or personnel required to perform a set of Shuttle functions (e.g., Avionics, SAIL, SSME controller).
System Integration	The process of combining hardware and software into a system and verifying its intended functions and interfaces.
Systematic Test Sequence	A logical progression of increasingly more involved tests with each level of testing being an extension of the previous test culminating in the verification of the computer system. The systematic test sequence also implies a three–step approach to testing: testing the feasibility of a design concept; testing of preliminary design; and verification of the final version.
Top–down Approach	Approach used in defining software test requirements from the system level down to the module or subroutine level. Sometimes used to define test sequence when starting with the operating system and pseudo or “dummy” modules.
Trace	A trace is an interpretive diagnostic technique which provides an analysis of each executed instruction resulting in a listing of such information as the contents of words and registers as they are modified and the order in which the instructions are performed.

Unit	Same as module.
Unit Testing	The testing of individual modules.
User	The organization that will utilize the system to accomplish its functional objective.
Verification (Software)	The process of ensuring that the end item product satisfies all design requirements and the means by which this is demonstrated.

THIS PAGE INTENTIONALLY LEFT BLANK